Consideren una tarea que require tomar caracteres desde un teclado y mostrarlos en la pantalla de un monitor. La tasa de transferencia desde un teclado a un computador está limitado por la velocidad del tipeo del usuario que por lo general no excede los pocos caracteres por segundo. La tasa de transferencia desde el computador al monitor es un poco mayor y está limitada por la velocidad en que los caracteres pueden ser transmitidos sobre el enlace entre el computador y el monitor. Esta tasa de transferencia es mucho menor que la velocidad del procesador. Esta diferencia de velocidad entre el procesador y los distintos dispositivos crea la necesidad de usar mecanismos para sincronizar sus operaciones.

Una posible solución a este problema es conocida como spin waiting. Por ejemplo para llevar a cabo una salida en el monitor, el CPU envía el primer caracter y espera por una señal del monitor que indique que el caracter ya ha sido mostrado. El CPU envía el segundo caracter y espera por una señal que indique que el segundo caracter fue mostrado. Estos pasos se repiten para cada caracter a ser transferido. En el caso de una entrada desde el teclado, un caracter es enviado. El CPU espera por una señal que le indique que una tecla ha sido presionada y que el código correspondiente está disponible en alguna localidad apropiada, y asi sucesivamente para el resto de los caracteres. Esta solución en donde el CPU espera explícitamente por Entrada/Salida ocasiona un desperdicio enorme de ciclos de CPU. El CPU chequea constantemente el estado del dispositivo y durante ese periodo, el procesador no lleva a cabo ningún trabajo provechoso. Otras tareas pudieran llevarse a cabo mientras el dispositivo de E/S lleva a cabo su trabajo.

Otra alternativa es un chequeo ocasional, en este caso el CPU debe recordar periódicamente que debe revisar los dispositivos. Si pregunta muy a menudo desperdicia mucho tiempo revisando y si toma mucho tiempo entre verificación entonces podemos tener dispositivos ociosos. Otra posible solución es el uso de las interrupciones. La idea es que

1. el CPU le indica al dispositivo que requiere un servicio
2. el CPU se pone a trabajar en otras tareas
3. el dispositivo interrumpe al CPU cuando ha completado el servicio
4. el CPU toma la información proporcionada por el dispositivo y
5. continua una vez obtenido el servicio deseado.

Desde el punto de vista del programa del usuario, una interrupción es precisamente eso, una interrupción en la secuencia normal de ejecución del programa. El programa del usuario no tiene que incluir ningún código para posibilitar las interrupciones. Se añade el ciclo de interrupción al ciclo de instrucción.

Con el uso de interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S está en curso

Aspectos a considerar en interrupciones

- Señal de petición de interrupción
- Procesamiento de una interrupción
- Rutina de servicio de la interrupción
- Vector de Interrupción
- Habilitación o deshabilitación de interrupciones
- Prioridad de interrupción
- Interrupciones no enmascarables

Señal de petición de Interrupción

La interrupción debe procesarse sin intervención del software que se está ejecutando en el momento en el que ocurre. Para esto es necesario que el hardware provea el mecanismo básico a través del cual se dé curso a la interrupción. Para esto es necesario que el CPU tenga una línea de entrada que provenga del dispositivo y que una señal que llegue por dicha línea obligue al CPU a atenderla.

Para permitir el uso de interrupciones, se añade un ciclo de interrupción al ciclo de instrucción (Figura 1)

En el ciclo de interrupción, el procesador comprueba si se ha producido alguna interrupción indicada por la presencia de una señal de interrupción. Si no hay señales pendientes de interrupción, el procesador continúa con el ciclo de captación y accede a la siguiente instrucción del programa en curso. Si hay una interrupción pendiente el procesador suspende la ejecución del programa en curso y guarda su contexto (dirección de la próxima instrucción, estado de los registros, PSW, etc.) y carga el Contador de Programa (PC) con la dirección de comienzo de la rutina de servicio de interrupciones. A continuación el procesador continúa con el ciclo de captación de la siguiente instrucción y de esta forma accede a la primera instrucción del manejador de interrupciones que dará servicio a la interrupción que ocurrió. Al finalizar la rutina de servicio de interrupciones, el control puede ser retornado al programa interrumpido y su ejecución continua a partir del punto donde fue suspendido.

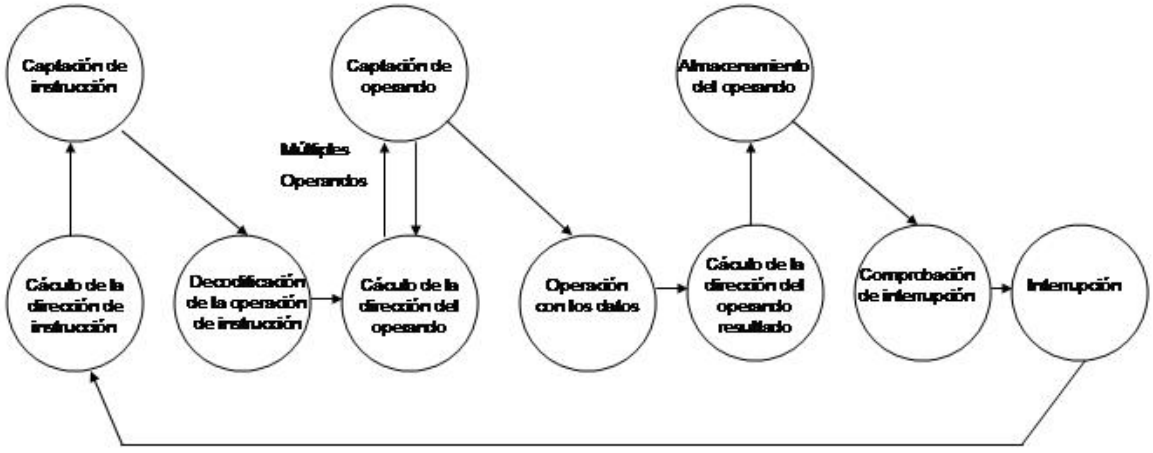


Figura 1. Ciclo de ejecución de una instrucción

Esta técnica supone cierta penalización u overhead pues deben ejecutarse instrucciones extras que permiten determinar el tipo de interrupción que ocurrió y decidir la acción apropiada. Pero esto es mejor que esperar a que se lleve a cabo una operación de E/S.

Este mecanismo de interrupción permite responder a estímulos externos (E/S) e interrumpir la ejecución de un programa. La ocurrencia de una interrupción es asíncrona pues puede ocurrir en cualquier momento de la ejecución del programa. Una interrupción puede ser vista como una llamada a una subrutina que no recibe argumentos y no devuelve ningún valor.

Procesamiento de una interrupción

Cuando un dispositivo termina una operación, se produce la siguiente secuencia de eventos en el hardware:

1. El dispositivo envía una señal de interrupción al procesador
2. El procesador termina la ejecución de la instrucción en curso antes de responder a la interrupción como se muestra en la figura 1
3. El procesador comprueba si hay interrupciones, determina que hay una, y envía una señal de reconocimiento al dispositivo que originó la interrupción. La señal de reconocimiento hace que el dispositivo desactive su señal de interrupción
4. En este instante el procesador necesita prepararse para transferir el control a la rutina de servicio de interrupción. Para empezar, debe guardar la información necesaria para continuar el programa en curso en el punto en que se interrumpió. La información mínima es el estado del procesador (PSW) y la dirección de la siguiente instrucción a ejecutar que está contenida en el contador de programa. Estos registros se pueden introducir en la pila de control del sistema.
5. Después, el procesador carga el contador del programa con la posición de inicio de la rutina de servicio de interrupciones. Según sea la arquitectura del computador y el diseño del sistema de operación, puede haber un solo programa manejador de interrupciones, o uno por cada tipo de interrupción, o uno por cada dispositivo y cada tipo de interrupción. Si hay más de una rutina manejadora, el

procesador debe determinar el tipo de interrupción ocurrida para llamar al programa asociado.

Rutina de Servicio de Interrupciones

Una vez que el contador de programa se ha cargado con la dirección de la rutina manejadora, el procesador continúa con el ciclo de instrucción y de esta forma se transfiere al programa manejador o de servicio de interrupciones. La ejecución de esta rutina da lugar a las siguientes operaciones:

6. Hasta este momento, se han almacenado en la pila del sistema el contador de programa y el PSW del programa interrumpido. Sin embargo, hay otra información que se considera estado del programa en ejecución. Se deben guardar los contenidos de los registros del procesador puesto que estos registros pueden ser utilizados por la rutina de interrupción
7. La rutina de interrupción puede continuar ahora procesando la interrupción.
8. Cuando el procesamiento de la interrupción finaliza, los valores de los registros almacenados se recuperan de la pila y se vuelven a almacenar en los registros.
9. El paso final es recuperar los valores del PSW y del contador del programa desde la pila. Como resultado de esto, la siguiente instrucción que se ejecute pertenecerá al programa previamente interrumpido.

Es importante almacenar toda la información del estado del programa interrumpido para que éste pueda reanudarse. Dado que el manejador de interrupciones puede ser ejecutado desde cualquier punto, no puede haber una preparación explícita como ocurre con las llamadas a las subrutinas. El manejador de interrupciones debe ser escrito cuidadosamente para asegurar que los registros usados por el programa interrumpido no se vean afectados. Por lo tanto, el manejador debe hacer todo el trabajo de salvado de registros pues puede ser invocado en cualquier punto y antes de finalizar debe recuperar los contenidos de los registros usados.

Vector de Interrupción

Uno de los pasos descritos anteriormente implica la búsqueda de la dirección de la rutina de servicio de la interrupción. Si bien las direcciones de las distintas rutinas podrían ser fijas y estar “cableadas” en hardware, existe una solución sencilla que permite mayor flexibilidad para definir las direcciones en la que se encuentran dichas rutinas. En esta solución lo que se fija en hardware son las direcciones en las que el usuario podrá escribir las verdaderas direcciones de las rutinas.

Por ejemplo en una arquitectura sencilla las localidades de memoria 100, 104 y 108 podrían estar reservadas para que contengan las direcciones de las rutinas de servicio de interrupción del teclado, del monitor y la impresora. Cuando ocurre una interrupción del monitor, el hardware buscará en la dirección 104 la dirección a la que debe saltar.

Habilitación o deshabilitación de Interrupciones

Un problema que puede ocurrir con la atención de las interrupciones es un anidamiento de interrupciones. La manera más fácil de enfrentar este problema es disponer de instrucciones que deshabiliten las interrupciones y antes de retornar de la rutina de servicio se deben habilitar nuevamente. A veces es necesario deshabilitar interrupciones cuando una rutina de servicio está actualizando estructuras de datos críticas que pueden ser accesadas desde otras rutinas de servicio de interrupciones. Otra forma para resolver el problema de la habilitación/deshabilitación de interrupciones es el de las prioridades

Prioridad de Interrupción

Un mecanismo más general para el control de las interrupciones anidadas consiste en asignar niveles de importancia a las distintas interrupciones y evitar que ocurra una interrupción de menor importancia durante la ejecución de una rutina de servicio de interrupción.

Además de evitar las interrupciones anidadas, este mecanismo permite asignar prioridades a los dispositivos de acuerdo al grado de urgencia que requiera su atención.

Interrupciones no enmascarables

Alguna interrupciones podrían estar definidas como de alta prioridad, de manera que no sea posible deshabilitarlas ni evitarlas por medio del mecanismo de prioridades. Un ejemplo sería la interrupción de una fuente de poder para indicar que sólo quedan unos minutos de energía eléctrica. Este tipo de interrupción debe proceder a como de lugar, salvando datos para evitar que quede información inconsistente en el disco.

Tipos de Excepciones o Interrupciones

El mecanismo de interrupciones es de uso más general. Hasta ahora lo hemos visto aplicado a E/S. Pero hay una variedad de razones por las cuales es deseable interrumpir la ejecución de un programa.

A parte de los dispositivos de E/S demandando un servicio, el Sistema de Operación puede interrumpir la ejecución de un proceso. Para esto, el HW provee generalmente un *timer* o temporizador que se activará cada cierto tiempo e interrumpe el programa retornando el control al Sistema de Operación. Otro uso de los mecanismos de interrupción es para ocuparse de condiciones extraordinarias que pueden ocurrir dentro de la ejecución normal de un proceso.

Por ejemplo cuando una instrucción aritmética produce un overflow o desbordamiento, es importante que esto sea reconocido para que el programa pueda hacer algo o que finalice con un mensaje apropiado de error. Es ineficiente que el programa explícitamente chequee si ocurrió un desborde cada vez que ejecuta una instrucción aritmética.

Esta situación puede ser manejada generando una excepción. En este caso se conoce como TRAP porque es el resultado directo de la ejecución de un programa y no depende de un evento externo. Es un evento síncrono que ocurre cada vez que se ejecute el programa usando los mismos datos, y este evento ocurrirá en el mismo sitio.

Otra razón para invocar el manejador de interrupciones o excepciones es si el programa intenta hacer algo no permitido o no definido. Por ejemplo, acceder direcciones de memoria fuera del rango permitido

Cómo se comunica un dispositivo de E/S ?

Un módulo de E/S es el elemento del computador responsable del control de uno o más dispositivos externos y del intercambio de datos entre esos dispositivos y la memoria principal y/o los registros del CPU. Asi, el módulo de E/S debe tener una interfaz interna al computador con el CPU y la memoria principal y una interfaz externa al computador con el dispositivo.

La complejidad de los módulos de E/S y el número de dispositivos externos que controlan varían considerablemente. En la figura 2 se muestra un diagrama de bloques de un módulo de E/S. El módulo se conecta al computador a través de un conjunto de líneas (bus). Los datos que se transfieren a y desde el módulo se almacenan temporalmente en uno o más registros de datos. Además puede haber uno o más registros de estado que proporcionan información del estado presente. Un registro de estado puede funcionar también como un registro de control para recibir información de control del CPU. La lógica que hay en el módulo interactúa con el CPU a través de una serie de líneas de control. Estas líneas son las que utiliza el CPU para proporcionar las órdenes al módulo de E/S

Para indicarle un comando a un dispositivo de E/S, el procesador debe ser capaz de direccionar el dispositivo y suplirle una o más instrucciones de comando. Cuando el CPU, la memoria y los dispositivos de E/S comparten un bus común, son posibles dos modos de direccionamiento :

- E/S asignada a memoria (Memory Mapped I/O)
- E/S aislada

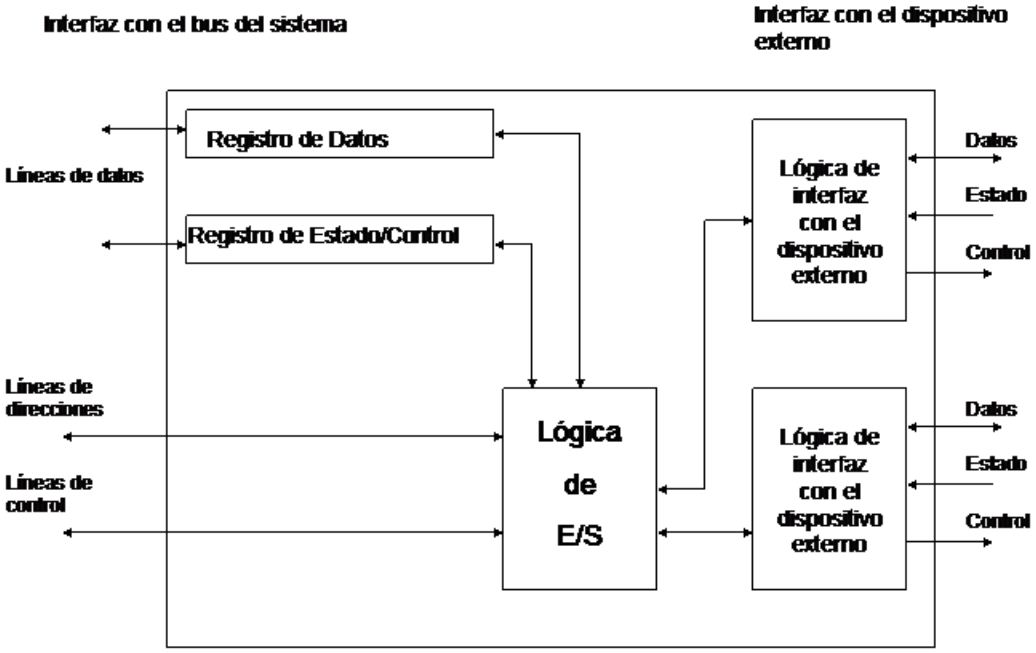


Figura 2. Diagrama de bloques de un módulo de E/S

Con la E/S asignadas a memoria, existe un único espacio de direcciones para las posiciones de memoria y los dispositivos de E/S. El CPU considera a los registros de estado y de datos de los módulos de E/S como posiciones de memoria y utiliza las mismas instrucciones para acceder tanto a memoria como a los dispositivos de E/S.

Manejo de dispositivos

El registro de Estado/Control asociado a un dispositivo generalmente contiene información que nos indica el estado de dicho dispositivo. En el registro de Estado/Control podemos encontrar los siguientes campos:

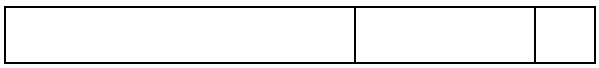
- READY: un bit que nos indica si el dispositivo está listo para recibir algún comando o que finalizó con el servicio requerido.

- Tipo de Servicio: También puede tener asociado una serie de bits que sirven para indicarle el tipo de servicio que se le está solicitando (Read, Write, etc).
- Interrup Enable: Bit que indica si el dispositivo está habilitado para generar interrupciones para el procesador
- Parámetros: bit para incluir parámetros necesarios para el servicio solicitado

Como mencionamos anteriormente, estos registros pueden estar en posiciones reservadas de memoria (E/S asignada a memoria) o pueden estar en registros especiales y son necesarias instrucciones especiales para acceder a dicha información.

Ejemplo I

Dado un sensor de temperatura que registra la temperatura e interrumpe al procesador si la temperatura detectada excede un cierto valor prefijado. Supongamos que este dispositivo se maneja asignado a memoria, es decir que los regsitros de datos y de Estado/Control que tiene asociados se encuentran en alguna dirección de memoria reservada (por ejemplo 0xFFFF1000). Queremos escribir un programa simple que permita habilitar la interrupción de este sensor cuando la temperatura exceda los 80°C



Interrupt (Interrupt enable)

Temp

Registro de Estado/Control del Sensor

```
define Sensor   0xFFFF1000                  #   Dispositivo asignado a memoria
VectorInterrupt[Sensor] = SensorManejador   #   Instalación de la rutina manejadora que
                                                se hará cargo de la interrupción que
                                                ocasione el Sensor
Sensor.Temp = 80                            #   Temperatura a la cual interrumpirá el
                                                sensor
Sensor.interrupt = ON                       #   habilitación de interrupciones

  .....
 #  aqui va cualquier código que deba ejecutar el procesador


SensorManejador()
{       se incluye cualquier acción que se quiera efectuar cuando el sensor detecte que la
        temperatura excedió los 80°C prefijados
        Si no queremos tener interrupciones anidadas mientras se ejecuta la rutina,
        entonces se deben deshabilitar momentaneamente las interrupciones por parte de
        este dispositivo o de todos los dispositivos que maneja el sistema.
        Sensor.interrupt = OFF
          .....     Acciones de la rutina


        Sensor.interrupt = ON
}
```

Noten que la rutina SensorManejador no es invocada en ningún momento. Cuando ocurra la interrupción ocasionada por el dispositivo Sensor, el procesador se dará cuenta de que existe una interrupción de algún dispositivo y tendrá que ver cuál es el dispositivo que está interrumpiendo. Una vez conocido cuál de los dispositivos interrumpió se ejecuta la rutina manejadora asociada con dicha interrupción (SensorManejador). Una vez finalizada la rutina manejadora se retorna a la siguiente instrucción en donde ocurrió la interrupción.

Ejemplo II

Supongamos que disponemos de un temporizador (TIMER) que interrumpe cada segundo. Se desea que implemente un reloj que lleve la hora del día.

```
VectorInterrup[TIMER] = TimerManejador
Reloj = 0 horas, 0 minutos, 0 segundos

.....

TimerManejador()
{
        segundos = segundos + 1
        if (segundos == 60 )
        {       segundos = 0
                minutos = minutos + 1
                if ( minutos == 60 )
                {       minutos = 0
                        horas = horas + 1
                        if ( horas == 24 )
                                horas = 0
                }
        }
}
```

Ejemplo III

Problema de la Escalera Mecánica.

Supongamos que una escalera mecánica posee una plancha en la entrada de la escalera que al ser pisada pñor un usuario activa el motor que pone a funcionar la escalera. Además dispone de otra plancha en la salida de la escalera. Cuando esta plancha es pisada se detiene el funcionamiento de la escalera mecánica.

Dispositivos presentes en el problema

Plancha de Entrada (dispositivo que interrumpe)

Plancha de Salida (dispositivo que interrumpe)

Motor de la escalera (dispositivo que no interrumpe)

Registro de Estado/Control de las Planchas

Enable: un bit que indica si la plancha puede interrumpir al procesador

Registro de Estado/Control del motor

Encendido : un bit que indica si el motor está encendido o apagado

Supongamos que los distintos dispositivos que intervienen en este problema están asignados a diversas direcciones de memoria

Plancha Entrada	0x FFFF1000
Plancha Salida	0x FFFF1004
Motor	0x FFFF1008

VectorInterrupt[Plancha Entrada] = PlanchaEntradaManejador

VectorInterrupt[Plancha Salida] = PlanchaSalidaManejador

Motor.Encendido = OFF

PlanchaEntrada.Enable=TRUE

PlanchaSalida.Enabke = TRUE

integer contador = 0

```
PlanchaEntradaManejador( )
{       contador ++
        Motor.Encendido = ON
}

PlanchaSalidaManejador( )
{       contador - -
        if (contador <= 0)
        {       Motor.Encendido = OFF
                contador = 0
         }
}
```

Esta versión no es correcta pues puede ocurrir lo que se conoce como Condición de Carrera pues estamos usando una variable que es compartida por ambos manejadores y si ocurren interrupciones anidades, su valor final puede ser incorrecto

Qué pasa si entra una madre con su hijo y ambos pisan la plancha de entrada, pero el niño sale de la escalera en los brazos de su madre ? El motor no se apaga.

Una posible solución es que tengamos sólo una plancha de entrada y un temporizador que va a permitir que el motor se apague después de un tiempo razonable para que el usuario de la escalera mecánica llegue a la plancha de salida.